Class #4

op art, curves, math
QUIZ

- gross
- ugh
- bleah
OP ART
libraries

- import > library > pdf
Render to PDF

• draws directly to a pdf file, instead of window

• can be used to create images much larger than screen

• size(600, 600, PDF,"line.pdf");
Render to screen, export to PDF

- draws both to screen and exports to pdf file
- not so great for giant files
import processing.pdf.*;

size(200, 200);
beginRecord(PDF, "processing_pdf.pdf");
fill(220, 23, 45);
rect(20, 20, 75, 87);
endRecord();
other options

• save one frame to pdf on command
• save multiple frames into one pdf
• save TIFF to off screen buffer
• save out as tiles
functions
Constrain

- \( x = \text{constrain}(y, 20, 100) \)
void

• functions which do not return a value are proceeded by `void`

```cpp
void drawRandomCircle() {
    int x = int(random(100));
    int y = int(random(100));
    ellipse(x, y, 50, 50);
}
```
passing parameters

• to send values to a function declare them with the function declaration

```c
void drawRandomCircle(int w, int h) {
  int x = int(random(100));
  int y = int(random(100));
  ellipse(x,y,w,h);
}```
return values

• some functions return a value
• declare data type of functions that return a value
• use the return statement to send data back to whatever called the function
Return Values

```java
void draw()
{
    int x = addit(5,6);
    println (x);
}

int addit(int x, int y) {
    int result = x + y;
    return result;
}
```
Beziers curves

- `bezier(x1, y1, cx1, cy1, cx2, cy2, x2, y2)`
  - `x1, y1` - 1st anchor point
  - `cx1, cy1` - 1st curve point
  - `cx2, cy2` - 2nd curve point
  - `x2, y2` - 2nd anchor point
some math

• sq(value) - square number and return result
• sqrt(value) square root of value and return result
• pow(number, exponent) raises number to exponent and return result
normalization

• convert numbers to a value between 0 and 1 for simplification of calculations

• norm(value, low, high)

• convert back to range by multiplying by original high value
curves

- use exponential functions to create simple curves
- $y = x^n$
- use normalization to keep $y$ less than 1